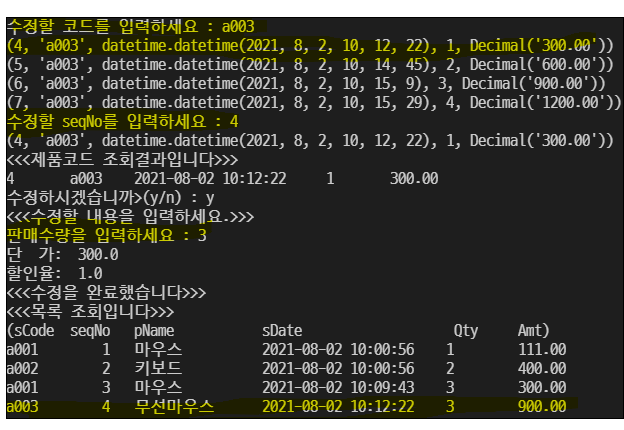
빅데이터분석결과시각화\_평가\_답안지

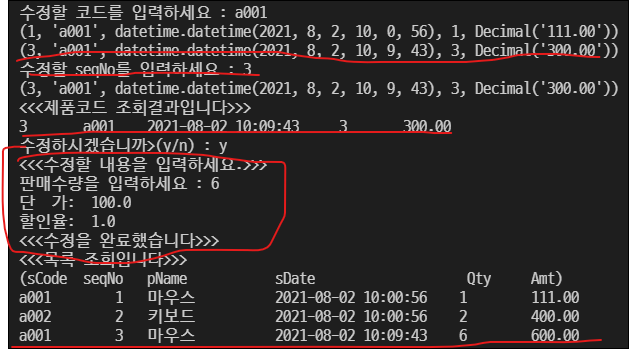
성명 : 전우영

작성일 : 2021년 08월 02일

<문제-1>

**다. 판매정보 수정 시 정상동작 확인**

1-1. 예를 들면 나.항에서 입력한 제품 중  a003 제품의 순번 4번의 정보를 보면 판매수량 1개이고 판매금액이 300입니다. 이 판매수량을 3으로 바꾸었을 때 판매금액이 900으로 변경하는지 확인하세요.  
  
1-2. 1-1항과 같은 판매수량변경을 임의 로 6개 이상 변경하고 수량변경에 따른 판매금액 변경을 확인하여 스샷하세요.



**라. 외부파일을 읽어서 판매테이블에 저장하세요**

2. sales\_2.txt파일을 읽어서 필터링

2-1 리스트 저장

2-2 리스트 내용 필터링

2-2-1 연월일 정보 오류 시 해당 자료는 판매테이블 저장대상에서 제외

2-2-2 제품코드가 product테이블에 존재하지 않는 자료는 판매테이블 저장대상에서 제외

2-2-3 판매수량을 테이블 product 테이블의 단가를 이용하여 산출 후 sales Table에 저장

**>> 실행코드**

import os

import sys

import pymysql

import datetime

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from datetime import datetime

from re import match,findall,sub

from re import findall, match, sub

#%matplotlib inline

pd.set\_option('display.notebook\_repr\_html', True)

pd.set\_option('display.max\_columns', 7)

pd.set\_option('display.max\_rows', 10)

pd.set\_option('display.width', 250)

config = {                  # MySQL데이터베이스를 사용하기 위해 환경변수를 딕셔너리로 생성함.

    'host' : '127.0.0.1',   # MySql 이 동작하는 ipv4주소

    'user' : 'root',        # MySql 설치할 때 정한 계정

    'passwd' : 'root1234',  # MySql 설치할 때 정한 비밀번호

    'database' : 'test\_db', # MySql 설치할 때 처음 생성한 데이터베이스

    'port' : 3306,          # MySql이 응용프로그램과 통신할 때 사용하는 포트번호 리터럴이 아닌 정수값으로 써야함

    'charset' : 'utf8',     # 한글을 사용하겠다는 설정

    'use\_unicode' : True    # 한글을 사용하겠다는 설정

    }

============================  [가] 판매등록 기능 ==============================# 기본로직 : 제품코드 유무를 시작으로 제품을 등록!

#   1. class ProductFind 중 def productfind() 메서드로 제품코드 존재여부 확인 --> True/False

#   1-1. (True) 판매 수량/총액 입력

#              - 판매수량 : salesInput()를 통해 입력값 검수(\*) 및 반환(\*\*)

#                \* 검수: InputFilter 클래스의 setQty 메서드 True/False반환

#                \*\* 반환: InputFilter 클래스의 setQty가 True이면 salesInput()함수에 입력값 반환

#              - 판매총액 : amtFunction()를 통해 product 테이블로부터 단가와 할인율 select하여 amt 값을 구하여 반환

#   1-2. (False) 제품등록 후 판매등록 진행

# [가] 판매등록 기능 메인함수: salesCreate()

======================================================================

def salesCreate() : # 판매등록

    #os.system('cls')

    try :

        conn = pymysql.connect(\*\*config)

        cursor = conn.cursor()

        #

        flag = 1

        while flag:

            #os.system('cls')

            print("<<<판매등록입니다>>>")

            in\_sCode = input("판매한 제품코드를 입력하세요 : ")  # 제품코드 입력

            if in\_sCode != '' :

                c1 = ProductFind\_sales(in\_sCode) #제품코드 유무 파악

# 2-1. product테이블에 제품코드가 존재하면 판매정보 입력

                if len(c1.productfind()) : # 제품코드 있으면 판매일자 및 수량을 기입하고 총액을 구하여 sales 테이블에 등록

                    qty = salesInput() # 판매수량 입력함수로` 이동

                    in\_Qty = int(qty)

                    in\_Amt = amtFunction(in\_Qty,in\_sCode) # 매출액 계산함수

                    sql = f"insert into sales(sCode,qty,amt) values ('{in\_sCode}',{in\_Qty}, {in\_Amt})"

                    cursor.execute(sql)

                    conn.commit()

                    print("판매등록을 성공했습니다.")

                    flag = 0 # 판매등록이 완성되면 메인으로 이동

                # 2-2. product 테이블에 제품코드가 존재하지 않으면 제품등록 등록 후 판매등록

                else :

                    print("존재하지 않는 제품입니다. 제품등록부터 진행하겠습니다.....")

                    num = int(input("1. 진행 OK 2. 진행 NO >>.."))

                    if num == 1 :

                        productCreate\_sales() # product 테이블 제품등록 함수

                        print("제품등록을 마쳤습니다.")

                        flag=1 # 제품등록이 마치면 다시 처음으로 돌아가 판매 제품코드 입력부터 시작

                    else:

                        print("제품등록을 하지 않습니다.")

                        flag=1

            else :

                print("판매등록을 위해 코드를 입력해 주세요.")

                flag=1

    except Exception as e :

        print('salesCreate() 오류 : ', e)

        conn.rollback()

    finally:

        cursor.close()

        conn.close()

    print("<<<목록 조회입니다>>>")

    sd = SalesFind(2,in\_sCode,'')

    rows = sd.salesfind()

    print("(sCode\tseqNo\tpName\t\tsDate\t\t\tQty\tAmt)")

    for row in rows :

        print("{:4}  {:5}\t{:7}  \t{}{:5}\t{:5}".format(row[0],row[1],row[2],row[3],row[4],row[5]))

##################################################################################

### (2021-08-02 월요일) txt파일 유효자료 DB에 등록하기

##################################################################################

def salesCreateExtra(data):

    print("외부파일 등록입니다.")

    in\_sCode = data[0]

    sdate = data[1]

    in\_sDate = sdate[0:4]+"-"+sdate[4:6]+"-"+sdate[6:]

    in\_sQty = data[2]

    try :

        conn = pymysql.connect(\*\*config)

        cursor = conn.cursor()

        #

        flag = 1

        while flag:

            #os.system('cls')

            print("<<<외부파일에 대한 판매등록입니다>>>")

            if in\_sCode != '' :

                c1 = ProductFind\_sales(in\_sCode) #제품코드 유무 파악

                # product테이블에 제품코드가 존재하면 판매정보 입력

                if len(c1.productfind()) : # 제품코드 있으면 판매일자 및 수량을 기입하고 총액을 구하여 sales 테이블에 등록

                    qty = in\_sQty

                    in\_Qty = int(qty)

                    in\_Amt = amtFunction(in\_Qty,in\_sCode) # 매출액 계산함수

                    sql = f"insert into sales(sCode, sDate, qty,amt) values ('{in\_sCode}','{in\_sDate}',{in\_Qty}, {in\_Amt})"

                    cursor.execute(sql)

                    conn.commit()

                    print("외부파일 정보 판매등록을 성공했습니다.")

                    flag = 0 # 판매등록이 완성되면 메인으로 이동

                # 2-2. product 테이블에 제품코드가 존재하지 않으면 제품등록등록 후 판매등록

                else :

                    print("존재하지 않는 제품입니다. 제품등록부터 진행하겠습니다.....")

                    num = int(input("1. 진행 OK 2. 진행 NO >>.."))

                    if num == 1 :

                        productCreate\_sales() # product 테이블 제품등록 함수

                        print("제품등록을 마쳤습니다.")

                        flag=1 # 제품등록이 마치면 다시 처음으로 돌아가 판매 제품코드 입력부터 시작

                    else:

                        print("제품등록을 하지 않습니다.")

                        flag=1

            else :

                print("판매등록을 위해 코드를 입력해 주세요.")

                flag=1

            #

    except Exception as e :

        print('salesCreateExtra() 오류 : ', e)

        conn.rollback()

    finally:

        cursor.close()

        conn.close()

    print("<<<목록 조회입니다>>>")

    sd = SalesFind(2,in\_sCode,'')

    rows = sd.salesfind()

    print("(sCode\tseqNo\tpName\t\tsDate\t\t\tQty\tAmt)")

    for row in rows :

        print("{:4}  {:5}\t{:7}  \t{}{:5}\t{:5}".format(row[0],row[1],row[2],row[3],row[4],row[5]))

===========================  [나] 판매조회 기능 ===============================

# 기본로직 : 제품코드로 판매를 조회!

# [나] 판매조회 기능 메인 클래스: salesRead

=======================================================================

class SalesRead :

    def \_\_init\_\_(self): # 생성자 : read\_sel : 코드/상품명/all ,

        self.read\_sql = "select s.sCode, s.seqNo, p.pName, s.sDate, s.Qty, s.Amt from sales s inner join product p on s.sCode=p.pCode where s.sCode ="

    def salesReadOne(self,code) : # 코드를 받아 옴

        try :

            conn = pymysql.connect(\*\*config)    # 딕셔너리 config를 인수로 사용하여 conn 객체를 만듬.

            cursor = conn.cursor()              # conn 객체로부터 cursor() 메소드를 호출하여 cursor 참조변수를 만듬.

            #

            rows = []

            #os.system('cls')

            in\_code = code

            sr = ProductFind\_sales(in\_code)

            if sr.productfind(): # 2-1 product 테이블에 제품코드 존재 시 sales 테이블 대상으로 진행

                sql = self.read\_sql + f"'{in\_code}' order by s.sDate DESC"

                cursor.execute(sql)

                rows = cursor.fetchall()

                if len(rows) > 0 : # 2-1 product 테이블에 제품코드가

                    print("===코드명 테이블 조회===")

                    print("(sCode\tseqNo\tpName\t\tsDate\t\t\tQty\tAmt)")

                    for row in rows :

                        print("{:4}\t{:5}\t{:5}\t{}{:5}\t{:5}".format(row[0],row[1],row[2],row[3],row[4],row[5]))

                else:

                    print("조회결과 입력한 코드번호: {}에 맞는 정보가 없습니다".format(in\_code))

            else: # 2-2 product 테이블에 제품코다가 존재하지 않을 경우 "에러메시지 출력"

                print("존재하지 않는 코드입니다.")

        except Exception as e :

            print('SalesRead 오류 : ', e)

            conn.rollback() # 실행 취소

        finally:

            cursor.close()

            conn.close() # >> 이동완료

    def salesReadAll(self) :

        #os.system('cls')

        print("<<<목록 조회입니다>>>")

        sr = SalesFind(2,'','')

        rows = sr.salesfind()

        print("===테이블 조회1===")

        print("(sCode\tseqNo\tpName\t\tsDate\t\t\tQty\tAmt)")

        for row in rows :

            print("{:4}\t{:5}\t{:5}\t{}{:5}\t{:5}".format(row[0],row[1],row[2],row[3],row[4],row[5]))

============================= [다] 판매수정 기능 =============================

# 기본로직 : 제품코드로 존재여부 파악, seqNo 존재여부 파악, 수정 등 순으로 접근

# [다] 판매수정 기능 메인 함수: salesUdate()

=======================================================================

def salesUpdate() : # 판매정보 수정

    #os.system('cls')

    print("<<<판매정보 수정입니다>>>")

    sd = SalesFind(2,'','')

    rows = sd.salesfind() # 전체판매정보 보여주기

    for row in rows :

        print("{}\t{}\t{}\t{}\t{}\t{}".format(row[0],row[1],row[2],row[3],row[4],row[5]))

    find\_code = FindFunction(1) # '1'은 코드찾기를 의미함

    in\_sCode = input('수정할 코드를 입력하세요 : ')

    rows\_code = find\_code.findCode(in\_sCode) # 입력 코드를 찾아서 해당 값을 튜플로 반환

    if rows\_code: # 반환값 주소를 할당받은 참조변수 rows\_code가 존재할 경우

        for row in rows\_code:

            print(row)

        in\_seqNo = input('수정할 seqNo를 입력하세요 : ')

        find\_seq = FindFunction(2) # '2'는 seqNo 찾기를 의미함

        rows\_seq = find\_seq.findSeqno(in\_seqNo) # 입력 일련번호를 찾아서 해당 값을 튜플로 반환

        if rows\_seq: # 반환값 주소를 할당받은 참조변수 rows\_code가 존재할 경우

            for row in rows\_seq:

                print(row)

                flag = 1 # 코드와 일련번호가 존재할 경우 수정작업을 진행

        else:

            print("입력하신 seqNo정보가 없습니다.")

    else:

        print("입력하신 코드정보가 없습니다.")

        flag = 0

    while flag: # 수정작업 진행

        up1 = SalesFind(4,in\_sCode,in\_seqNo) # '4'의 의미는 판매수정 작업번호를 의미

        rows = up1.salesfind()

        #

        try :

            conn = pymysql.connect(\*\*config)

            cursor = conn.cursor()

            if rows :

                print("<<<제품코드 조회결과입니다>>>")

                for row in rows :

                    print("{}\t{}\t{}\t{}\t{}".format(row[0],row[1],row[2],row[3],row[4]))

                yesNo = input('수정하시겠습니까>(y/n) : ')

                if yesNo == "y" or yesNo == "Y":

                    print("<<<수정할 내용을 입력하세요.>>>")

                    qty = salesInput()

                    in\_Qty = int(qty)

                    in\_Amt = amtFunction(in\_Qty,in\_sCode)

                    sql = f"update sales set Qty = {in\_Qty}, Amt={in\_Amt} where sCode = '{in\_sCode}' and seqNo = '{in\_seqNo}'"

                    cursor.execute(sql)

                    conn.commit()

                    print("<<<수정을 완료했습니다>>>")

                    flag=0

                else:

                    print("<<<수정을 취소했습니다.>>>")

                    flag=0

            else :

                print('수정할 코드가 없습니다.')

                flag=0

                pass

        except Exception as e :

            print('salesUpdate() 오류: ', e)

            conn.rollback() # 실행 취소

        finally:

            cursor.close()

            conn.close()

        print("<<<목록 조회입니다>>>")

        sd = SalesFind(2,in\_sCode,'')

        rows = sd.salesfind()

        print("(sCode\tseqNo\tpName\t\tsDate\t\t\tQty\tAmt)")

        for row in rows :

            print("{:4}\t{:5}\t{:5}\t{}{:5}\t{:5}".format(row[0],row[1],row[2],row[3],row[4],row[5]))

    print("판매정보 수정을 마치겠습니다.")

==============================  [라] 판매삭제 기능 ============================

# 기본로직 : 제품코드로 존재여부 파악, seqNo 존재여부 파악, 수정 등 순으로 접근

# [다] 판매수정 기능 메인 함수: salesUdate()

=======================================================================

def salesDelete() : # 판매정보 삭제

    #os.system('cls')

    print("<<<판매정보 삭제입니다>>>")

    sd = SalesFind(2,'','')

    rows = sd.salesfind()

    for row in rows :

        print("{}\t{}\t{}\t{}\t{}\t{}".format(row[0],row[1],row[2],row[3],row[4],row[5]))

    find\_code = FindFunction(1) # '1'은 코드찾기를 의미함

    in\_sCode = input('삭제할 코드를 입력하세요 : ')

    rows\_code = find\_code.findCode(in\_sCode)

    if rows\_code:

        for row in rows\_code:

            print(row)

        in\_seqNo = input('삭제할 seqNo를 입력하세요 : ')

        find\_seq = FindFunction(2) # '2'는 seqNo 찾기를 의미함

        rows\_seq = find\_seq.findSeqno(in\_seqNo)

        if rows\_seq:

            for row in rows\_seq:

                print(row)

                flag = 1

        else:

            print("입력하신 seqNo정보가 없습니다.")

    else:

        print("입력하신 코드정보가 없습니다.")

        flag = 0

    while flag:

        up1 = SalesFind(4,in\_sCode,in\_seqNo) # '4'의 의미는 판매수정 작업번호를 의미

        rows = up1.salesfind()

        #

        try :

            conn = pymysql.connect(\*\*config)

            cursor = conn.cursor()

            if rows :

                print("<<<제품코드 조회결과입니다>>>")

                for row in rows :

                    print("{}\t{}\t{}\t{}\t{}".format(row[0],row[1],row[2],row[3],row[4]))

                yesNo = input('삭제하시겠습니까>(y/n) : ')

                if yesNo == "y" or yesNo == "Y":

                    sql = f"delete from sales where seqNo = '{in\_seqNo}'"

                    cursor.execute(sql)

                    conn.commit()

                    print("<<<삭제 완료했습니다>>>")

                    flag=0

                else:

                    print("<<<삭제 취소했습니다.>>>")

                    flag=0

            else :

                print('삭제할 코드가 없습니다.')

                flag=0

                pass

        except Exception as e :

            print('salesDelete() 오류: ', e)

            conn.rollback() # 실행 취소

        finally:

            cursor.close()

            conn.close()

        print("<<<목록 조회입니다>>>")

        sd = SalesFind(2,in\_sCode,'')

        rows = sd.salesfind()

        print("(sCode\tseqNo\tpName\t\tsDate\t\t\tQty\tAmt)")

        for row in rows :

            print("{:4}\t{:5}\t{:5}\t{}{:5}\t{:5}".format(row[0],row[1],row[2],row[3],row[4],row[5]))

    print("판매정보 삭제를 마치겠습니다.")

# [가] ~ [라]에 적용된 함수/클래스 및 메서드

##################################################################################

### FindFunction 함수: 코드와 seqNo으로 정보찾기

##################################################################################

class FindFunction:

    def \_\_init\_\_(self, sel):

        self.sel = sel

        if self.sel == 1:

            self.sql = "select \* from sales where sCode ="

        elif self.sel ==2:

             self.sql = "select \* from sales where seqNo ="

    def findCode(self, code): #코드로 정보를 찾고 dataset 참조변수에 할당

        in\_code = code

        try:

            conn = pymysql.connect(\*\*config)

            cursor = conn.cursor()

            sql = self.sql + f"'{in\_code}'"

            cursor.execute(sql)

            dataset = cursor.fetchall()

        except Exception as e:

            print("findCode error: ",e)

            conn.rollback()

        finally:

            cursor.close()

            conn.close()

        return dataset

    def findSeqno(self, seqno): #Seqno로 정보를 찾고 dataset 참조변수에 할당

        in\_seqNo = seqno

        try:

            conn = pymysql.connect(\*\*config)

            cursor = conn.cursor()

            sql = self.sql + f"'{in\_seqNo}'"

            cursor.execute(sql)

            dataset = cursor.fetchall()

        except Exception as e:

            print("findSeqno error: ",e)

            conn.rollback()

        finally:

            cursor.close()

            conn.close()

        return dataset

#######################################################################################################

### amtFunction 함수 : 코드로 product 테이블의 할인율과 단가를 select하고 수량을 곱하여 총액을 구함

#######################################################################################################

def amtFunction(in\_Qty,in\_sCode): # 총매출과 제품명을 product테이블을 활용하여 판매등록과 수정에 활용

    # ---- amt = qty \* unitprice \* discountrate ----

    in\_unitPrice, in\_discountRate = 0,0

    try:

        conn = pymysql.connect(\*\*config)

        cursor = conn.cursor()

        # product 테이블에서 제품코드(in\_pCode)를 통해서 제품명, 단가, 할인율을 받아옴

        sql = f"select UnitPrice, discountRate from product where pCode = '{in\_sCode}'"

        cursor.execute(sql)

        rows = cursor.fetchall()

        in\_unitPrice = float(rows[0][0]) # 단가

        in\_discountRate = float(rows[0][1]) # 할인율

        #print("단  가: ", in\_unitPrice)

        #print("할인율: ", in\_discountRate)

        conn.commit()

    except Exception as e:

        print("salesCrete 중 단가/할인율 error: ", e)

        conn.rollback()

    finally:

        cursor.close()

        conn.close()

    in\_Amt = in\_Qty \* in\_unitPrice \* in\_discountRate

    return in\_Amt

#######################################################################################################

### ProductFind 함수 : product 테이블에서 정보를 select 하여 튜플형으로 반환

#######################################################################################################

class ProductFind\_sales :

    def \_\_init\_\_(self,find\_in\_data=''):

        self.find\_sql = "select \* from product where pCode =" + f"'{find\_in\_data}'"

    def productfind(self) :

        try :

            conn = pymysql.connect(\*\*config)    # 딕셔너리 config를 인수로 사용하여 conn 객체를 만듬.

            cursor = conn.cursor()              # conn 객체로부터 cursor() 메소드를 호출하여 cursor 참조변수를 만듬.

            cursor.execute(self.find\_sql)

            return cursor.fetchall()    # select 쿼리문의 실행 결과를 return함

                                        # 쿼리의 실행결과가 없으면 요소의 갯수가 0인 리스트가 반환됨

        except Exception as e :

            print('ProductFind\_sales 오류 : ', e)

            conn.rollback() # 실행 취소

        finally:

            cursor.close()

            conn.close()

#################################################################################

### SalesFind 함수 : sales 테이블에서 정보를 select 하여 튜플형으로 반환

##################################################################################

class SalesFind :

    def \_\_init\_\_(self,find\_sel,find\_in\_sCode='',find\_in\_seqNo=''):

        self.read\_sel = find\_sel

        if find\_sel == 3 or find\_sel == 4 or find\_sel == 5:

            self.find\_sql = f"select \* from sales where sCode ='{find\_in\_sCode}' and seqNo = '{find\_in\_seqNo}'"

        elif find\_sel == 2:

            self.find\_sql = f"select s.sCode, s.seqNo, p.pName, s.sDate, s.Qty, s.Amt from sales s inner join product p on s.sCode=p.pCode order by s.sDate"

        else :

            self.find\_sql = "select s.sCode, s.seqNo, p.pName, s.sDate, s.Qty, s.Amt from sales s inner join product p on s.sCode=p.pCode where sCode ='{find\_in\_sCode}' oder by s.sDate"

    def salesfind(self) :

        try :

            conn = pymysql.connect(\*\*config)    # 딕셔너리 config를 인수로 사용하여 conn 객체를 만듬.

            cursor = conn.cursor()              # conn 객체로부터 cursor() 메소드를 호출하여 cursor 참조변수를 만듬.

            cursor.execute(self.find\_sql)

            return cursor.fetchall()    # select 쿼리문의 실행 결과를 return함

                                        # 쿼리의 실행결과가 없으면 요소의 갯수가 0인 리스트가 반환됨

        except Exception as e :

            print('SalesFind 오류 : ', e)

            conn.rollback() # 실행 취소

        finally:

            cursor.close()

            conn.close()

##################################################################################

### salesInput() 함수 : sale 테이블에 입력할 Qty(판매수량)를 필터링하고 in\_qty 참조변수에 값의 주소를 반환

##################################################################################

def salesInput(): # 수량입력 함수

    si1 = InputFilter()

    while True:

        # string으로 받음

        if si1.setQty(input("판매수량을 입력하세요 : ")) : # 판매수량을 문자로 받아서 숫자만 입력하도록 함

            in\_qty = si1.qty # 입력값이 문제 없으면 in\_qty에 할당

            break

        else:

            continue

    return in\_qty # 입력값을 반환

##################################################################################

### productCreate\_sales() 함수 : 판매등록 시 제품코드가 없을 경우 진행

##################################################################################

def productCreate\_sales() : # 제품등록

    try :

        conn = pymysql.connect(\*\*config)

        cursor = conn.cursor()

        #

        #os.system('cls')

        print("<<<제품 등록입니다>>>")

        in\_pCode = input("등록할 제품코드를 입력하세요 : ")  #

        if in\_pCode != '' :

            c1 = ProductFind\_sales(in\_pCode)

            if len(c1.productfind()) :

                print("이미 존재합니다.")

            else :

                iValue = userInput\_sales(in\_pCode)

                #

                sql = f"insert into product(pCode, pName, UnitPrice, discountRate) values('{iValue[0]}','{iValue[1]}', '{iValue[2]}', '{iValue[3]}')"

                cursor.execute(sql)

                rows = cursor.fetchall()

                conn.commit()

                print("제품등록을 성공했습니다.")

                print()

        else :

            print("제품등록을 위해 코드를 입력해 주세요")

    except Exception as e :

        print('productCreate\_sales() 오류 : ', e)

        conn.rollback()

    finally:

        cursor.close()

        conn.close()

##################################################################################

### userInput(): productCreate\_sales()) 함수 진행시 각 컬럼의 값을 입력

##################################################################################

def userInput\_sales(in\_pCode):

    code = in\_pCode

    ui1 = InputFilter()

    # 제품입력

    while True:

        if ui1.setPname(input("제품명을 입력하세요 : ")) :

            in\_pname = ui1.pname

            break

        else:

            continue

    # 제품코드 입력

    while True:

        userinputCode = ui1.setPcode(code)

        if userinputCode :

            in\_pcode = ui1.pcode

            break

        else:

            code = input("제품코드를 다시 입력: ")

            continue

    # 제품가격 입력

    while True:

        if ui1.setUnitPrice(input("제품가격을 입력하세요 : ")) :

            in\_unitPrice = ui1.unitPrice

            break

        else:

            continue

    # 할인율 입력

    while True:

        if ui1.setDiscountRate(input("제품 할인율을 입력하세요 : ")) :

            in\_discountRate = ui1.discountRate

            break

        else:

            continue

    #

    print("userInput>> ", in\_pname)

    return in\_pname,  in\_pcode, in\_unitPrice, in\_discountRate

##################################################################################

### InputFilter 클래스 : 입력값이 올바르게 input이 되었는지 필터링하고, 이상 유무에 따라 True/False 반환

##################################################################################

class InputFilter :

    def \_\_init\_\_(self):

        self.inputValueFilter\_result = False

    # [판매:Sales] 판매수량 검사

    def setQty(self, qty) : # string형 qty

        sale = int(qty)

        if sale == 0 or sale < 0 :

            print("0 이거나 음수일때 "+qty+" 의 정보는 부정확합니다.")

            self.inputValueFilter\_result = False

        elif sale>0:

            #st\_sale = str(sale)

            cnt=0

            for i in range(len(qty)):

                if ord("0") <= ord(qty[i]) <= ord("9"):

                    cnt += 1

            if cnt >= len(qty):

                self.inputValueFilter\_result = True

                self.qty = sale

        return self.inputValueFilter\_result

    # [공통:prouct&sales] 코드 검사

    def setPcode(self, pcode) :

        ls = pcode

        if ls == '' or len(ls) < 0 :

            print("공백일때 "+str(pcode)+" 의 정보는 부정확합니다.")

            self.inputValueFilter\_result = False

        else:

            self.inputValueFilter\_result = True

            self.pcode = pcode

        return self.inputValueFilter\_result

    # [제품:prouct] 이름 검사

    def setPname(self, pname) :

        ls = pname

        if ls == '' or len(ls) < 0 :

            print("공백일때 "+str(pname)+" 의 정보는 부정확합니다.")

            self.inputValueFilter\_result = False

        else:

            self.inputValueFilter\_result = True

            self.pname = pname

        return self.inputValueFilter\_result

    # [제품:prouct] 가격 검사

    def setUnitPrice(self, price) :

        if price == '' or len(price) < 0 :

            print("공백일때 "+str(price)+" 의 정보는 부정확합니다.")

            self.inputValueFilter\_result = False

        else:

            self.inputValueFilter\_result = True

            self.unitPrice= price

        return self.inputValueFilter\_result

    # [제품:prouct] 할인율검사

    def setDiscountRate(self, discountRate) :

        if discountRate == '' or len(discountRate) < 0 :

            print("공백일때 "+str(discountRate)+" 의 정보는 부정확합니다.")

            self.inputValueFilter\_result = False

        else:

            self.inputValueFilter\_result = True

            self.discountRate= discountRate

        return self.inputValueFilter\_result

#

def tableCreate\_sales() :

    try :

        conn = pymysql.connect(\*\*config)

        cursor = conn.cursor()

        sql = """create table sales (

            seqNo int(10) not null auto\_increment,  # 자동증가하는 순번

            sCode varchar(4) not null,

            sDate timestamp default now(),

            Qty int not null,

            Amt decimal(12, 2) default 0.0,

            primary key(seqNo,sCode))"""

        cursor.execute(sql)

        conn.commit()

    except Exception as e :

        print("tableCreate() 오류 : ",e)

        conn.rollback()

    finally :

        conn.close()

        cursor.close()

#

##################################################################################

### (2021-08-02 월요일) txt 파일에서 유효한(저장 가능한) 데이터를 뽑아내기

##################################################################################

# step 01: data폴더의 txt 파일을 찾아서 파일 내용을 dataset 리스트에 취합

def fileReadRecord():

    try:

        txt\_data = "data/" # 경로지정

        file\_list = os.listdir(txt\_data) # txt\_data 목록반환

        for fname in file\_list:

            file\_path = txt\_data+fname

            if os.path.isfile(file\_path):

                with open(file\_path, mode = "r", encoding = "utf8") as ft:

                    rows = ft.readlines()

                    dataset = []

                    for row in rows:

                        line = row.strip()

                        dataset.append(line.split(',')) # 리스트 형으로 각 요소를 dataset 리스트에 할당

            else:

                print("파일이 존재하지 않습니다.")

    except Exception as e:

        print("openfile error : ", e)

    finally:

        pass

    return dataset

# step 02: dataset 리스트를 받아서 각 리스트형 요소를 뽑아서 CheckList 클래스에 보내고 Boolin 형으로 받아서 최종 data(fianlData)를 생성

def makeDataFromFile(dataset):

    rawData = dataset

    cl = CheckList()

    cnt = 0

    finalData = []

    for raw in rawData:

        cnt +=1

        print(f"{cnt}차 검사시작 하겠습니다.")

        print("코드는 ",raw[0], "이며, 판매일자는",raw[1], "입니다.")

        chk\_code = cl.checkProductCode(raw[0])

        chk\_date = cl.checkProductSalesDate(raw[1])

        if chk\_code == True and chk\_date == True:

            finalData.append(raw)

        else:

            pass

    return finalData

# step 03: makeDataFromFile()함수로부터 받은 rawData의 코드와 판매일자를 검수

class CheckList:

    def \_\_init\_\_(self):

        self.inValueResult = False

    def checkProductCode(self, code): # fileReadRcorde() 함수에서 받은 파일을 체크

        self.inPcode = code

        c1 = ProductFind\_sales(self.inPcode) # ProductFind\_sales 클래스를 통해서 존재여부 파악

        if len(c1.productfind()) :

            print("ChekList 통지: 존재합니다.")

            self.inValueResult = True # 존재하면 True 반환

        else :

            print("ChekList 통지: 존재하지 않습니다.")

            self.inValueResult = False # 존재하지 않으면 False 반환

        return self.inValueResult

    def checkProductSalesDate(self,date): # 판매일자 체크

        self.inSdate = date

        if len(self.inSdate) == 8:

            year = int(self.inSdate[0:5]) #txt의 판매일자에서 앞의 4자리 연도수 값의 주소를 year 참조변수에 할당

            #print(datetime.now())

            now\_date = datetime.now() # 현재 기준 날짜를 값의 주소를 now\_date 참조변수에 할당 / 목적은 연도는 해마다 증가하기 때문임.

            #now\_date = 2021

            month = int(self.inSdate[5:7]) # txt의 판매일자에서 월 값의 주소를 month 참조변수에 할당

            day = int(self.inSdate[7:]) # txt의 판매일자에서 일 값의 주소를 day 참조변수에 할당

            if 1920<= year <= now\_date.year :

            #if 1920<= year <= now\_date : #now\_date.year: # 1920년 이후부터 올해까지 연도 범위에 있는지 검사

                isLeapYear = (year%4 and year%100 !=0) or (year%400 == 0) # 윤년/평년을 구분 윤년일 경우 1(True)을 반환하고 아닐 경우 0(False)를 반환

                days = [31,28+isLeapYear,31,30,31,30,31,31,30,31,30,31] # 윤년/평년에 따라 2월의 마지막 날 28일에 1 혹은 0을 더함

                if 1<= month <=12:

                    if 1<=day<=days[month+1]: # 연,월,일 모든 조건문을 만족 시 True 값을 반환

                        self.inValueResult = True

                    else:

                        self.inValueResult = False

                else:

                    self.inValueResult = False

        else:

            print("ChekList 통지: 잘못된 날짜입니다.")

            self.inValueResult = False # 잘못 입력 시 False 반환

        return self.inValueResult

##################################################################################

if \_\_name\_\_ == "\_\_main\_\_" :

    tableCreate\_sales()

    while True:

        #os.system('cls')

        print("---제품관리---")

        print("판매    등록 : 1 ")

        print("판매목록조회 : 2 ")

        print("코드별  조회 : 3 ")

        print("판매    수정 : 4 ")

        print("판매    삭제 : 5 ")

        print("판매관리종료 : 9 ")

        sel = int(input("작업을 선택하세요 : "))

        if sel == 1 :

            num = int(input("1. 직접등록 2. 외부파일 등록 >> "))

            if num == 1:

                print("1-1. 직접등록입니다.")

                salesCreate()

            elif num ==2:

                print("1-2. 외부파일 등록입니다.")

                dataset = fileReadRecord()

                finalData = makeDataFromFile(dataset)

                for final in finalData:

                    salesCreateExtra(final)

            print("작업을 완료했습니다.")

            os.system("pause")

        elif sel == 2 :

            r2 = SalesRead()

            r2.salesReadAll()

            os.system("pause")

        elif sel == 3 :

            r3 = SalesRead()

            r3.salesReadOne(input("코드를 입력하세요>> "))

            os.system("pause")

        elif sel == 4 :

            salesUpdate()

            os.system("pause")

        elif sel == 5 :

            salesDelete()

            os.system("pause")

        elif sel == 9 :

            print("판매관리를 종료합니다. ")

            os.system("pause")

            os.system('cls')

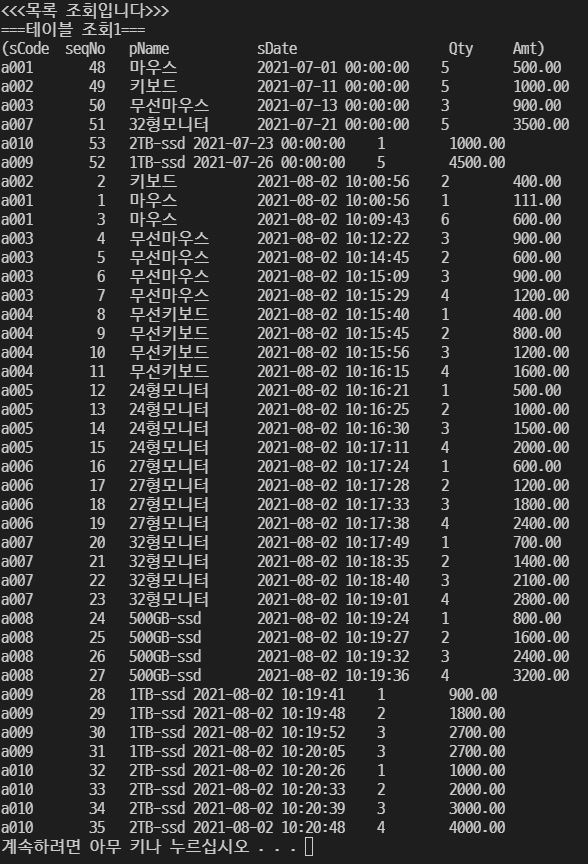
            sys.exit(0)

        else :

            print("잘못 선택했습니다. ")

            os.system("pause")

**>> 실행결과**

****

**마. 시각화 결과물을 도출하세요.**  
**1. sales\_all.txt 파일로 만드세요**

**>> 실행코드**

##################################################################################

### 마 1. sales\_all.txt파일 만들기 ( 선택항목 6번 파일생성 함수 )

##################################################################################

def makeSalesFile():

    print("salesAll 파일 생성")

    try:

        conn = pymysql.connect(\*\*config)

        cursor = conn.cursor()

        sql = "select \* from sales"

        cursor.execute(sql)

        rows = cursor.fetchall()

        #print(rows)

        rawData = []

        sr\_list = []

        ls = []

        for line in rows:

            for st in line:

                sr\_list.append(str(st))

            ls.append(sr\_list)

            sr\_list=[]

        #print(ls)

        ls2=[]

        for st in ls:

            string = "|".join(st)

            ls2.append(string)

        #print(ls2)

        salesFile = "\n".join(ls2)

        print(salesFile)

        file\_path = "data/sales\_all.txt"

        with open(file\_path, mode = "w", encoding = "utf8") as ft:

            ft.write(salesFile)

    except Exception as e:

        print("makeSalesFile error: ", e)

        conn.rollback()

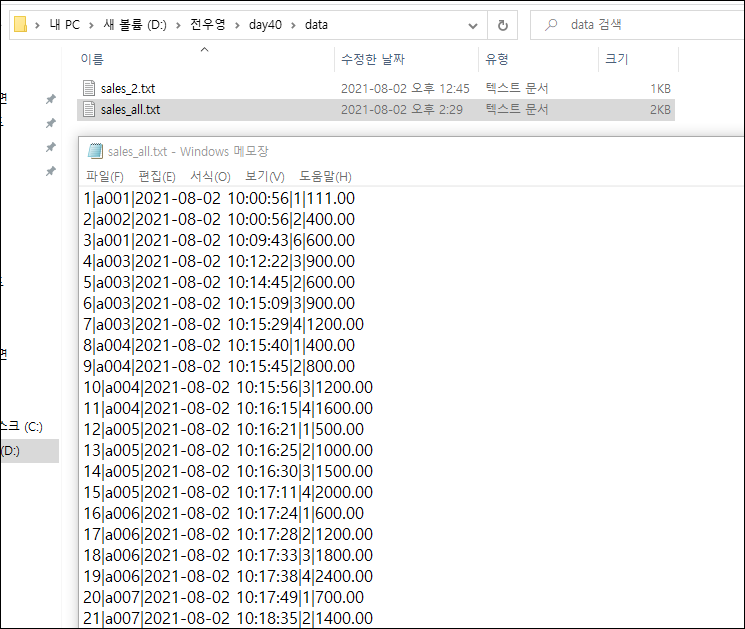
    finally:

        cursor.close()

        conn.close()

##########################################################################################################################################################

>> 실행결과



**2. sales\_all.txt파일을 읽어서 DataFrame으로 만들기**

**2-1. 인덱스는 날짜, 제품코드 중 자유롭게 지정**

**>> 실행코드**

##################################################################################

### 마 2. sales\_all.txt파일을 읽고 DataFrame으로 만들기

##################################################################################

def makeSalesDataFrame(): # sales\_all.txt파일을 읽고 DataFrame으로 만들기

    print("DataFrame 만들기>>")

    try:

        file\_path = "data/sales\_all.txt"

        with open(file\_path, mode="r", encoding="utf8") as ft:

            salesFile = ft.readlines()

            ls = []

            for i in salesFile: # sales\_all.txt 파일을 ls list형으로 담기

                ls.append(i.strip())

    except Exception as e:

        print("makeSalesDataFrame 함수 error e: ", e)

    finally:

        pass

    ls\_seqNo = [] # 일련번호 리스트

    ls\_code = [] # 판매제품 리스트

    ls\_index = [] # 인덱스 리스트 (일련번호 + 판매제품)

    ls\_date = [] # 판매일자 리스트

    ls\_int\_qty = [] # 판매수량 리스트

    ls\_int\_amt = [] # 판매금액 리스트

    cnt = 0

    for factor in ls: # sales\_all.txt 파일 내용이 담긴 ls 리스트 요소를 불러오기

        cnt = factor.find("|", cnt) # 파일 내 seqNo를 ls\_seqNo에 첨가하기

        ls\_seqNo.append(factor[:cnt]) #

        cnt\_code = factor.find("|", cnt+1) # 파일 내 code를 ls\_code에 첨가하기

        ls\_code.append(factor[cnt+1:cnt\_code])

        index = factor[:cnt] + factor[cnt+1:cnt\_code] # 파일 내 index를 ls\_index에 첨가하기

        ls\_index.append(index)

        cnt\_date = factor.find("|", cnt\_code+1) # 파일 내 date를 ls\_date에 첨가하기

        ls\_date.append(factor[cnt\_code+1:cnt\_date])

        cnt\_qty = factor.find("|", cnt\_date+1) # 파일 내 qty를 ls+qty에 첨가하기

        ls\_int\_qty.append(int(factor[cnt\_date+1:cnt\_qty]))

        ls\_int\_amt.append(float(factor[cnt\_qty+1:])) # 파일 내 qty를 ls\_int\_qty에 첨가하기

    sr\_index = pd.Series(ls\_index) # index 정보를 담은 list를 Pandas의 Series형으로 변환

    sr\_seqNo = pd.Series(ls\_seqNo, index=sr\_index) # seqNo 정보를 담은 list를 Pandas의 Series형으로 변환

    sr\_code = pd.Series(ls\_code, index=sr\_index) # code정보를 담은 list를 Pandas의 Series형으로 변환

    sr\_date = pd.Series(ls\_date, index=sr\_index)  # date정보를 담은 list를 Pandas의 Series형으로 변환

    sr\_int\_qty = pd.Series(ls\_int\_qty, index=sr\_index) # qty정보를 담은 list를 Pandas의 Series형으로 변환

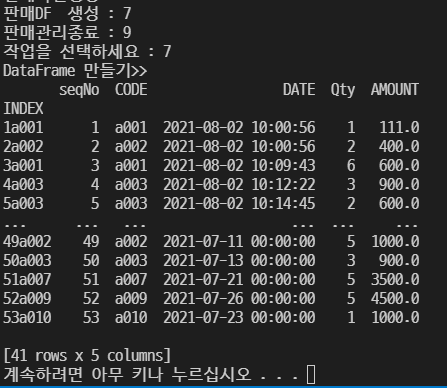
    sr\_int\_amt = pd.Series(ls\_int\_amt, index=sr\_index) # amt정보를 담은 list를 Pandas의 Series형으로 변환

    salesDataFrame = pd.DataFrame({"seqNo":sr\_seqNo, "CODE":sr\_code, "DATE":sr\_date, "Qty":sr\_int\_qty, "AMOUNT":sr\_int\_amt}) # 모든Series를 DataFrame으로 변환

    salesDataFrame.index.name='INDEX'

    print(salesDataFrame)

**>> 실행결과**



**3. 시각화 결과물**

**3-1. 날짜별 판매금액 :**

* **7월1일부터 8월 2일까지의 판매정보중 제품구분 없이 날짜별 총 합계를보여줌**

**3-2. 모델별 판매금액 :**

* **10개의 모델별로 모든 판매금액 합계를 보여줌**

>>실행코드

def visibleSalesDataFrame(dataframe):

    print("데이터 시각화")

    df=dataframe # 생성된 dataframe을 df 참조변수에 할당

    date = df["DATE"] # "DATE" 컬럼값 주소를 date 참조변수에 할당

    dateStart = input("시작 연도를 입력하시오: ")

    dateEnd = input("마지막 연도를 입력하시오: ")

    ask = (date > dateStart) & (date <= dateEnd) # date 참조변수를 기준으로 원하는 날짜를 각 로우에 Ture/False를 반환

    #print(ask)

    #print(type(ask))

    filtered\_df=df.loc[ask]

    sumAmt = filtered\_df["AMOUNT"].sum()

    sumQty = filtered\_df["Qty"].sum()

    print(f"1. >> {dateStart} ~ {dateEnd} 기간동안 발생한 총금액은 {sumAmt} 입니다.")

    print(f"2. >> {dateStart} ~ {dateEnd} 기간동안 발생한 총판매량은 {sumQty} 입니다.")

    groupCode = filtered\_df.groupby(['CODE'],as\_index = False).sum()

    print(groupCode)

>> 실행결과

